Energy-Efficient Resource Management for Federated Edge Learning With CPU-GPU Heterogeneous Computing

Qunsong Zeng, *Graduate Student Member, IEEE*, Yuqing Du, *Student Member, IEEE*, Kaibin Huang, *Fellow, IEEE*, and Kin K. Leung, *Fellow, IEEE*

***Abstract*— Edge machine learning involves the deployment of learning algorithms at the network edge to leverage massive distributed data and computation resources to train *artificial intelligence* (AI) models. Among others, the framework of *federated edge learning* (FEEL) is popular for its data-privacy preservation. FEEL coordinates global model training at an edge server and local model training at devices that are connected by wireless links. This work contributes to the energy-efficient implementation of FEEL in wireless networks by designing joint *computation-and-communication resource management* (C^2RM). The design targets the state-of-the-art heterogeneous mobile architecture where parallel computing using both CPU and GPU, called *heterogeneous computing*, can significantly improve both the performance and energy efficiency. To minimize the sum energy consumption of devices, we propose a novel C^2RM framework featuring multi-dimensional control including bandwidth allocation, CPU-GPU workload partitioning and speed scaling at each device, and C^2 time division for each link. The key component of the framework is a set of equilibriums in energy rates with respect to different control variables that are proved to exist among devices or between processing units at each device. The results are applied to designing efficient algorithms for computing the optimal C^2RM policies faster than the standard optimization tools. Based on the equilibriums, we further design energy-efficient schemes for device scheduling and greedy spectrum sharing that scavenges "spectrum holes" resulting from heterogeneous C^2 time divisions among devices. Using a real dataset, experiments are conducted to demonstrate the effectiveness of C^2RM on improving the energy efficiency of a FEEL system.**

***Index Terms*— Radio resource management, energy-efficient computation and communication, scheduling, federated learning, heterogeneous computing.**

I. Introduction

REALIZING the vision of exploiting the enormous data distributed at edge devices (e.g., smartphones and sensors) to train *artificial intelligence* (AI) models has been pushing machine learning from the cloud to the network edge, called *edge learning* [1]. Currently, arguably the most popular edge learning framework is federated learning that preserves users' privacy by distributed learning over devices [2]–[4]. Each *round* of the iterative learning process involves the broadcasting of a global model to devices, their uploading of local model updates computed using local datasets, and a server's aggregation of the received local updates for updating the global model. Executing complex learning tasks on energy and resource constrained devices is a main challenge faced in the implementation of edge learning. To tackle the challenge, the next-generation mobile *systems-on-chip* (SoC) will feature a heterogeneous architecture comprising a *central processing unit* (CPU) and a *graphics processing unit* (GPU) [and even a *digital processing unit* (DSP) in some designs] [5]. Experiments have demonstrated its advantages in terms of performance and energy efficiency. In this work, we address the issue of energy-efficient implementation of *federated edge learning* (FEEL) in a wireless system comprising next-generation devices capable of heterogeneous computing. To this end, a novel framework is proposed for energy-efficient joint management of *computation-and-communication* (C^2) resources at devices.

A. Edge Implementation of Federated Learning

The implementation of FEEL in wireless networks faces two key challenges among others. As mentioned, the distributed learning process requires potentially many edge devices to periodically upload high-dimensional local model updates to an edge server. This includes the first challenge that the excessive communication overhead generated in the learning process can overwhelm the air interface that has finite radio

Manuscript received November 2, 2020; revised April 2, 2021 and June 7, 2021; accepted June 7, 2021. Date of publication June 21, 2021; date of current version December 10, 2021. The work of Qunsong Zeng and Kaibin Huang was supported in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019B1515130003, in part by the Hong Kong Research Grants Council under Grant 17208319 and Grant 17209917, in part by the Innovation and Technology Fund under Grant GHP/016/18GD, and in part by the Shenzhen Science and Technology Program under Grant JCYJ20200109141414409. The work of Kin K. Leung was supported in part by the U.S. Army Research Laboratory and in part by the U.K. Ministry of Defence under Grant W911NF-16-3-0001. This article was presented in part at the IEEE International Conference on Communications Workshops, Dublin, Ireland, June 2020. The associate editor coordinating the review of this article and approving it for publication was X. Chen. *(Corresponding author: Kaibin Huang.)*

Qunsong Zeng, Yuqing Du, and Kaibin Huang are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong (e-mail: qszeng@eee.hku.hk; yqdu@eee.hku.hk; huangkb@eee.hku.hk).

Kin K. Leung is with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2BT, U.K. (e-mail: kin.leung@imperial.ac.uk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2021.3088910.

Digital Object Identifier 10.1109/TWC.2021.3088910

resources but needs to support FEEL as well as other services. Designing techniques for suppressing the overhead forms a vein of active research, called *communication-efficient FEEL*. Diversified approaches have been proposed such as device scheduling [6], [7], designing customized multi-access technologies [2], [3], and optimizing uploading frequencies [4].

The second challenge faced by FEEL is how to execute power hungry learning tasks (e.g., training AI models each typically comprising millions of parameters) on energy constrained devices. Tackling the challenge by designing energy-efficient techniques leads to the emergence of an active research theme, called *energy-efficient FEEL*, which is the topic of current investigation. While there exists a rich literature of energy-efficient techniques for *resource management* (RM) in radio access networks (see e.g., [8]), the designs of their counterparts for FEEL, which are the main research focus in energy-efficient FEEL, are different due to the changes on the system objective and operations. In particular, FEEL systems aim at improving the learning performance instead of providing a radio access service and performing update aggregation instead of decoupling multiuser data. In early works [9]–[11], researchers proposed radio-RM techniques (e.g., bandwidth allocation and device scheduling) to improve the tradeoff between devices' transmission energy consumption and learning performance. More recent research accounts for computation energy consumption which usually constitutes a substantial part of a device's total consumption in the learning process, motivating the design of C^{2}RM techniques in [12], [13]. In a standard FEEL design, the updates by devices are usually assumed synchronized. It arises from the operation of gradient aggregation and refer to the requirement that all local updates need to be received by the server before the global model can be updated. Consequently, all devices are allowed the same duration (per-round latency). In [12], the learning latency is measured by a weighted sum of individual devices' latencies and then the learning-energy tradeoffs are optimized. On the other hand, the idea of clock frequency control, or called *dynamic voltage and frequency scaling* (DVFS) [14], was explored in [13] for energy-efficient FEEL based on the assumption that each device has a CPU featuring DVFS. Then the C^{2}RM was optimized where the communication RM is based on either *time-division multiple access* (TDMA) or *non-orthogonal multiple access* (NOMA) and the computation RM is based on dynamic frequency scaling. While prior work assumes single-processor devices, CPU-GPU heterogeneous computing discussed in the sequel is a new paradigm of mobile computing supporting applications with intensive data crunching. The area of energy-efficient FEEL based on heterogeneous computing is uncharted and explored in this work.

B. CPU-GPU Heterogeneous Computing

The mentioned next-generation heterogeneous SoC are capable of supporting diversified workloads such as communication, signal processing, inference, and learning, which arise from a wide range of new mobile applications. Examples of such chips include Snapdragon by Qualcomm, R-series by AMD, and Kirin 970 by Huawei. Via the cooperation of the CPU and GPU on the same chip for executing a single task, *heterogeneous computing* on such SoC fully utilizes the computation resources of both processors to optimize the computation performance and energy efficiency [15]. In particular, the new paradigm has been demonstrated by experiments to accelerate the running of deep neural networks on smartphones [16], [17]. Existing research on heterogeneous computing focuses on several main design issues including workload partitioning [5], [14], [18], [19], dynamic frequency scaling [20], [21], and memory access scheduling [22]. The first two issues are addressed in this work. Specifically, workload partitioning refers to dividing and allocating workload of a task over the integrated CPU and GPU according to the task requirements and the processors' states (e.g., temperatures) and characteristics (e.g., speeds and energy efficiencies) so as to optimize the overall performance and efficiency [5], [14], [18], [19]. On the other hand, frequency scaling (or DVFS) previously considered for a single CPU [13] becomes the more sophisticated management in heterogeneous computing due to the joint CPU-GPU control [21]. While prior work on heterogeneous computing focuses on a single device, we study the joint control of workload partitioning and DVFS in the context of a large system comprising multiple devices and furthermore explore their integration with radio-RM.

C. Contributions

The objective of this work is not to contribute any new learning technique but to focus on C^{2}RM to facilitate the implementation of a standard FEEL technique in a wireless network. To this end, we consider a FEEL system consisting of one edge server and multiple edge devices. Its main difference from those in existing work (see e.g., [12]) is that each device is equipped with a CPU-GPU platform enabling heterogeneous computing. The design objective is to minimize the sum energy consumption at devices under a guarantee on learning performance (latency and accuracy) by jointly controlling C^{2}RM in the following four dimensions:

1) *Bandwidth allocation*: Allocating bandwidths to devices for transmission of local model updates under a constraint on the total uplink bandwidth;
2) *C^2 time division*: Dividing the allowed per-round latency for the computation and communication of each device;
3) *CPU-GPU workload partitioning*: Partitioning and allocating the computation workload for each device to its CPU and GPU;
4) *CPU-GPU frequency scaling*: Controlling the CPU-GPU frequencies/speeds at each device.

While the RM control in the first two dimensions are considered in prior work as discussed, the other two are unique for heterogeneous computing. Their joint control over multiple devices is a challenging and open problem. To the best of the authors' knowledge, this work represents the first attempt on studying heterogeneous computing in the context of energy-efficient FEEL. The main contributions are described as follows.

- **Equilibrium based C^{2}RM framework:** It is mathematically proved that the control policies are optimal if and only if they achieve the following equilibriums:
  1) The CPU-GPU pair of every device has *equal energy-workload rates*, defined as the increase in energy consumption per additional workload for each processing unit;
  2) A similar equilibrium exists with respect to the processing units' computation speeds as their optimal values are proved to be proportional to the corresponding workloads.
  3) All devices have *equal energy-time rates*, defined as the energy rates with respect to the communication/computation latency;
  4) All devices have *equal energy-bandwidth rates*, defined as the energy rates with respect to the allocated bandwidth.

  The equilibriums are applied to obtaining the optimal polices for 1) computation RM, 2) communication RM, and 3) joint C^{2}RM. They are either derived in closed-form or computed using low-complexity algorithms. In particular, for low-complexity joint C^{2}RM, the problem is decomposed into one master problem for achieving the equilibrium in energy-time rates, and two sub-problems for separately achieving the two equilibriums in energy-workload rates and energy-bandwidth rates. The resultant algorithm is shown to have much lower complexity than a standard solution method such as *block coordinate descent* (BCD).
- **C^2 aware scheduling:** Building on the equilibrium framework, an energy-efficient scheduling scheme is designed to select a fixed number of devices for participating in FEEL. The novelty lies in the design of *C^2 aware scheduling metric* that balances both the channel state and computation capacity of each device. Specifically, given the objective of sum energy minimization, the metric is designed to be the energy consumption of a device given the derived optimal C^2 time division and equal bandwidth allocation. In addition, we analyze the effect of the number of scheduled devices on model convergence.
- **Greedy spectrum sharing:** For the preceding designs, we assume fixed bandwidth allocation in each round of the iterative learning process. Consequently, the heterogeneous computation latencies of devices generates "spectrum holes" (unused spectrum-time blocks) that can be scavenged for further improving the energy efficiency. To this end, the scheme of greedy spectrum sharing is designed featuring a novel metric, called *energy-bandwidth acceleration rate* and defined as the derivative of the energy-bandwidth rate, for selecting an available device to transmit using the spectrum hole. A larger value of the metric, implies steeper energy reduction for a device when it is allocated additional bandwidth.

II. Models and Metrics

Consider the FEEL system in Fig. 1 comprising a single edge server and K edge devices, denoted by an index set $\mathcal{K} = \{1, \cdots, K\}$. Each iteration between local gradient uploading and global model updating is called a *communication round*, or *round* for short. It is assumed that the server has perfect knowledge of the multiuser channel gains and local computation characteristics, which can be obtained by feedback. Using this information, the server determines the energy-efficient strategies for device scheduling and C^{2}RM.

A. Federated Learning Model

A standard federated learning technique (see e.g., [23]) is considered as follows. A global model, represented by the parameter set $\mathbf{w}$, is trained collaboratively across the edge devices by leveraging local labelled datasets. For device k, let $\mathcal{D}_k$ denote the local dataset and define the local loss function as $F_k(\mathbf{w}) = \frac{1}{|\mathcal{D}_k|}\sum_{(\mathbf{x}_j, y_j)\in\mathcal{D}_k} \ell(\mathbf{w}; \mathbf{x}_j, y_j)$, where $\ell(\mathbf{w}; \mathbf{x}_j, y_j)$ is the sample-wise loss function quantifying the prediction error of the model $\mathbf{w}$ on the training sample $\mathbf{x}_j$ with regard to its label y_j. For convenience, we assume a uniform size for local datasets: $|\mathcal{D}_k| \equiv D$, for all k. Then the global loss function on all the distributed datasets can be written as

$$F(\mathbf{w}) = \frac{\sum_{(\mathbf{x}_j, y_j)\in\cup_k\mathcal{D}_k} \ell(\mathbf{w}; \mathbf{x}_j, y_j)}{|\cup_k \mathcal{D}_k|} = \frac{1}{K}\sum_{k=1}^{K} F_k(\mathbf{w}). \quad (1)$$

The learning process is to minimize $F(\mathbf{w})$, that is, $\mathbf{w}^* = \arg\min F(\mathbf{w})$. For ease of exposition, we focus on the gradient-averaging implementation while the current designs also apply to the model-averaging implementation [23]. In each round, say the i-th round, the server broadcasts the current model $\mathbf{w}^{(i)}$ as well as selection indicators $\{\rho_k\}$ to all edge devices, where the indicator $\rho_k = 1$ if device k is scheduled, or 0 otherwise. Suppose M devices are scheduled for participation in each round, denoted by an index set $\mathcal{M}_i$ for the i-th round. Based on the received model $\mathbf{w}^{(i)}$, each scheduled device calculates the gradient $\nabla F_k(\mathbf{w}^{(i)})$ using its local dataset. Upon completion, the local gradients are transmitted to the server for aggregation:

$$\mathbf{g}^{(i)} = \frac{1}{M}\sum_{k\in\mathcal{M}_i} \nabla F_k(\mathbf{w}^{(i)}) \quad (2)$$

Synchronized updates by devices are assumed. The global model is then updated by *stochastic gradient descent* (SGD) as $\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta\mathbf{g}^{(i)}$, where η is the learning rate. The process iterates until the model converges.

B. Model of Heterogeneous Computing

1) Workload Model: Adopting a standard model (see e.g., [24]), the total workload W for a computation task is given as $W = N_{\mathsf{FLOP}} \times D$, where D is the local dataset size and N_{FLOP} denotes the number of *floating point operations* (FLOPs) needed for processing each sample. Furthermore, we define $f_k^{(c)}$ and ${f'}_k^{(c)}$ (in cycle/s) as the clock frequency of the CPU and GPU at device k, respectively. It follows that the computing speeds of CPU and GPU can be defined as

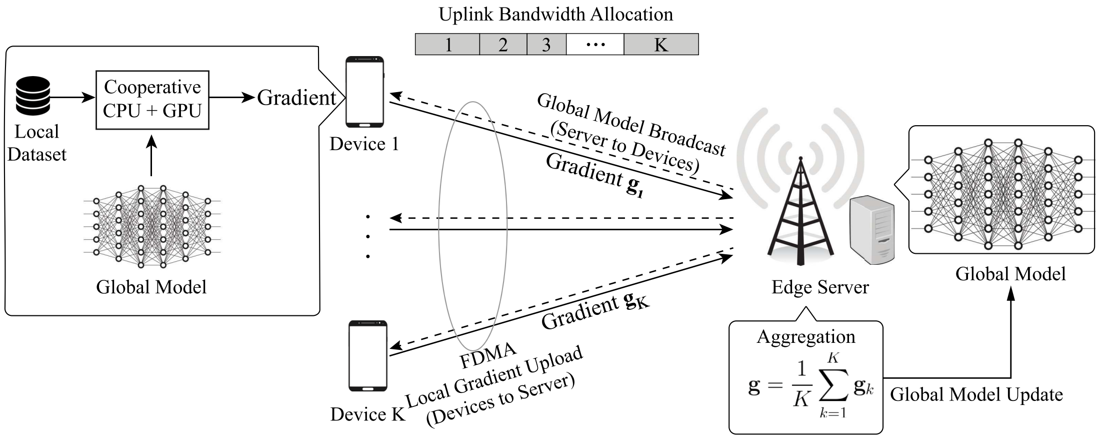


Fig. 1. A wireless system supporting FEEL over devices capable of CPU-GPU heterogeneous computing.

$f_k = f_k^{(c)} \times n_k$ and $f'_k = {f'}_k^{(c)} \times n'_k$ with n_k and n'_k denoting the number of CPU and GPU FLOPs per cycle, respectively.[1]

2) Workload Partitioning Model: For workload partitioning in local gradient computation, we consider input-sample partitioning, also known as *data parallelism* [27]–[29]. As a result, each processor (CPU or GPU) processes a fraction of input samples. As specified in [27], the local dataset $\mathcal{D}_k$ at device k is first partitioned into two sub-datasets, and then the CPU and GPU are assigned with the sub-datasets $\mathcal{D}_k^{\text{CPU}}$ and $\mathcal{D}_k^{\text{GPU}}$ satisfying $\mathcal{D}_k^{\text{CPU}} \cup \mathcal{D}_k^{\text{GPU}} = \mathcal{D}_k$ and $\mathcal{D}_k^{\text{CPU}} \cap \mathcal{D}_k^{\text{GPU}} = \emptyset$. The partitioned workloads for CPU and GPU at device k are denoted as W_k and ${W'}_k$ respectively, where $W_k = N_{\mathsf{FLOP}} \times |\mathcal{D}_k^{\text{CPU}}|$ and $W'_k = N_{\mathsf{FLOP}} \times |\mathcal{D}_k^{\text{GPU}}|$, with

$$\text{(Workload constraint)} \quad W_k + W'_k = W, \ \forall k \in \mathcal{M}. \tag{3}$$

In the training process, CPU and GPU independently calculate their gradients based on their assigned datasets in parallel, and then obtain $\mathbf{g}_k^{\text{CPU}} = \frac{1}{|\mathcal{D}_k^{\text{CPU}}|}\sum_{(\mathbf{x}_j,y_j)\in\mathcal{D}_k^{\text{CPU}}} \nabla\ell(\mathbf{w};\mathbf{x}_j,y_j)$ and $\mathbf{g}_k^{\text{GPU}} = \frac{1}{|\mathcal{D}_k^{\text{GPU}}|}\sum_{(\mathbf{x}_j,y_j)\in\mathcal{D}_k^{\text{GPU}}} \nabla\ell(\mathbf{w};\mathbf{x}_j,y_j)$. The local gradient of device k can be calculated by $\mathbf{g}_k = \frac{1}{D}\left(|\mathcal{D}_k^{\text{CPU}}|\mathbf{g}_k^{\text{CPU}} + |\mathcal{D}_k^{\text{GPU}}|\mathbf{g}_k^{\text{GPU}}\right)$, which gives the gradient $\nabla F_k(\mathbf{w})$. Given the partitioned workloads and CPU/GPU frequencies, the local computation time for the total workload W, denoted as ${t'}_k$, is determined by the processor that needs more time to finish completing the assigned workload and given as

$$\text{(Local computation time)} \ t'_k = \max\left\{\frac{W_k}{f_k}, \frac{W'_k}{f'_k}\right\}, \quad \forall k \in \mathcal{M}. \tag{4}$$

3) DVFS Model: For a CMOS circuit, the power consumption of a processor can be modeled as a function of clock frequency: $P = \Psi\left[f^{(c)}\right]^3$ with the coefficient Ψ [in Watt/(cycle/s)^3] depending on the chip architecture and $f^{(c)}$ being the clock frequency [30], [31]. Using this model,[2] the power consumption of CPU and GPU at device k can be written as[3]

$$\begin{aligned} P_k^{\text{CPU}} &= \Psi_k^{\text{CPU}}\left(f_k^{(c)}\right)^3 = C_k f_k^3, \\ P_k^{\text{GPU}} &= \Psi_k^{\text{GPU}}\left({f'}_k^{(c)}\right)^3 = G_k {f'}_k^3, \end{aligned} \tag{5}$$

where $C_k = \Psi_k^{\text{CPU}}/n_k^3$ and $G_k = \Psi_k^{\text{GPU}}/{n'}_k^3$. Using these functions, frequency scaling refers to controlling the power of CPU and GPU by adjusting their speeds f_k and f'_k, respectively.

Remark 1 (CPU/GPU Computation Efficiency): The coefficient C_k (or G_k) in (5) characterizes the *computation efficiency* of a CPU (or GPU), defined as the rate of power growth in response to the increase of the cubed computing speeds. In practical data processing based on heterogeneous computing, the GPU tends to play a main role (with $G_k < C_k$) while the CPU contributes supplementary computing resources.

4) Energy Consumption Model: Given the time durations W_k/f_k and W'_k/f'_k for CPU and GPU to complete their tasks with the workloads W_k and W'_k, the resultant energy consumption at device k can be written as $E_k^{\text{CPU}} = C_k W_k f_k^2$ and $E_k^{\text{GPU}} = G_k W'_k {f'}_k^2$. Then the total computation energy consumption of device k per round is

$$E_k^{\text{cmp}} = C_k W_k f_k^2 + G_k W'_k {f'}_k^2, \quad \forall k \in \mathcal{M}. \tag{6}$$

C. Multiple-Access Model

Consider the *frequency-division multiple access* (FDMA) for gradient uploading with the total bandwidth B. Let B_k denote the allocated bandwidth for device k in an arbitrary round, which is fixed throughout the round. This assumption is relaxed in Section V. Then we have the constraint:

$$\text{(Bandwidth constraint)} \quad \sum_{k\in\mathcal{M}} B_k = B. \tag{7}$$

[1] The architecture of a GPU comprises multiple *streaming multiprocessors* (SMs), each of which has many *streaming processors* (SPs) used in a single-instruction multiple data execution style [25]. The throughput of GPU satisfies $n'_k = N_k^{\text{SM}} \times \bar{n}'_k$ where N_k^{SM} and $\bar{n}'_k$ denote the number of SMs and the throughput per SM, respectively [26].

[2] The power consumption of the GPU is proportional to the number of SMs [26], that is $P_k^{\text{GPU}} = N_k^{\text{SM}} \times \Psi_k^{\text{SM}}\left({f'}_k^{(c)}\right)^3$, giving that $\Psi_k^{\text{GPU}} = N_k^{\text{SM}} \times \Psi_k^{\text{SM}}$. Therefore, the GPU model in [31] is applicable.

[3] For neural network training, the gradient is calculated using *backpropagation* (BP) algorithm, which is a computation-intensive task [32]. Most of the operations in BP algorithm involve matrix multiplication [33] where the power consumption of GPU memory is negligible compared with GPU cores [34].

Channels are assumed to be frequency non-selective. The edge server usually recruits idling devices as workers that either static or at most moving at pedestrian speeds [23]. For this reason, we adopt the model of slow block fading. To be specific, the channel gain of device k, denoted as h_k, is assumed to remain unchanged within one round but varies *independently and identically* (i.i.d.) over rounds. Given synchronous updates, a time constraint is set for each round:

$$\text{(Latency constraint)}\quad {t'}_k + t_k \le T, \quad \forall k \in \mathcal{M}, \tag{8}$$

where t'_k and t_k denote the time for local training/computation and gradient uploading of device k, respectively; T is the maximum total time for one round. A prerequisite for a scheduled device is that it can meet the above time constraint. Let P_k^{TX} denote the transmission power [in Watt/Hz] of device k. The achievable rate can be written as $r_k = B_k \log_2\left(1 + \frac{P_k^{\text{TX}} h_k^2}{N_0}\right), \forall k \in \mathcal{M}$, where N_0 is the spectrum density of the complex white Gaussian channel noise. Let $L = |\mathbf{g}|\alpha$ denote the gradient size (in bit) with α denoting a sufficient large number of bits for quantizing each parameter with negligible distortion. Then the data rate is $r_k = L/t_k, \forall k \in \mathcal{M}$. By combining the results, the communication energy consumption of device k per round is

$$E_k^{\text{cmm}} = B_k P_k^{\text{TX}} t_k = \frac{B_k t_k N_0}{h_k^2}\left(2^{\frac{L}{B_k t_k}} - 1\right), \quad \forall k \in \mathcal{M}. \tag{9}$$

D. Performance Metric

It is assumed that the data distributed at devices satisfy the condition for model convergence within a finite number of rounds, denoted as N (see e.g., [35]). As implied by (9), capacity achieving codes are deployed to ensure reliable transmissions. Consequently, wireless channels have no effects on N though they affect per-round latencies and energy consumption. The objective of energy-efficient RM is to minimize the total energy consumption of all active devices in the N-round learning process, namely $\sum_{i=1}^{N}\sum_{k=1}^{K}\rho_k[i]\left(E_k^{\text{cmp}}[i] + E_k^{\text{cmm}}[i]\right)$ with E_k^{cmp} and E_k^{cmm} given in (6) and (9), respectively. Given block fading and the per-round latency constraint in (8), the objective can be straightforwardly proved to be equivalent to minimize the total energy consumption for each round. This is aligned with a standard approach in adaptive transmission and its proof is similar to those in the literature (see Lemma 1 in [36]) and thus omitted for brevity. It follows that for subsequent designs, it is sufficient to consider an arbitrary round and the corresponding total energy consumption, i.e., $\sum_{k=1}^{K}\rho_k\left(E_k^{\text{cmp}} + E_k^{\text{cmm}}\right)$, as the performance metric, is called *sum energy* hereafter.

III. Energy-Efficient C^2 Resource Management

In this section, considering the case that all devices are scheduled for uploading (i.e., $\rho_k = 1, \forall k \in \mathcal{K}$), we study: 1) computation RM, 2) communication RM, and 3) joint C^{2}RM by analyzing and computing the optimal policies. Then, the results are applied to establishing an energy-learning tradeoff.

A. Computation Resource Management

1) Problem Formulation: The computation resources distributed at devices can be managed by controlling their workload partitioning and speed scaling to minimize the sum computation energy. The design is formulated as the following optimization problem.

$$\begin{aligned}
(\mathbf{P1})\quad \min_{\{W_k, f_k, f'_k\}} \quad & \sum_{k=1}^{K}\left(C_k W_k f_k^2 + G_k W'_k f_k^{'2}\right)\\
\text{s.t.}\quad & W_k + {W'}_k = W, \quad W_k \ge 0,\ {W'}_k \ge 0,\\
& \forall k \in \mathcal{K},\\
& t'_k = \max\left\{\frac{W_k}{f_k}, \frac{W'_k}{f'_k}\right\}, \quad \forall k \in \mathcal{K},\\
& 0 \le t'_k \le {T'}_k, \quad \forall k \in \mathcal{K},
\end{aligned}$$

where the three sets of constraints follow from (3), (4), and the time constraint with T'_k being the allowed maximum computation time for device k.

2) Optimal Policy: The optimal policy for computation RM is derived in closed-form below.

Lemma 1 (Speed Scaling Rule): To minimize the sum computation energy, the computation speeds of CPU and GPU, f_k and f'_k, should be scaled by following

$$\frac{W_k}{f_k} = \frac{W'_k}{f'_k}, \quad \forall k \in \mathcal{K}, \tag{10}$$

given an arbitrary workload pair (W_k, W'_k) with $W = W_k + W'_k$.

Proof: See Appendix A. □

The optimal strategy of workload-proportional speed scaling in Lemma 1 is intuitive and results from equalizing CPU and GPU computation time to avoid the slower one becoming the bottleneck of local gradient computation. Then, with Lemma 1, the original Problem **P1** can be rewritten as follows:

$$\begin{aligned}
(\mathbf{P2})\quad \min_{\{W_k, t'_k\}} \quad & \sum_{k=1}^{K}\frac{C_k W_k^3 + G_k W_k^{'3}}{t_k^{'2}}\\
\text{s.t.}\quad & W_k + W'_k = W,\ W_k \ge 0, \quad W'_k \ge 0,\ \forall k \in \mathcal{K},\\
& 0 \le t'_k \le T'_k, \quad \forall k \in \mathcal{K}.
\end{aligned}$$

Next, it is found that the optimal policy achieves an equilibrium between the CPU and GPU on each device, in terms of *energy-workload rate*.

Lemma 2 (Energy-Workload Rate Equilibrium): Consider the *energy-workload rates* of CPU and GPU given as

$$\frac{\partial E_k^{\text{CPU}}}{\partial W_k} = \frac{3 C_k W_k^2}{t_k^{'2}} \quad \text{and} \quad \frac{\partial E_k^{\text{GPU}}}{\partial W'_k} = \frac{3 G_k W_k^{'2}}{t_k^{'2}}, \quad \forall k \in \mathcal{K}, \tag{11}$$

where $E_k^{\text{CPU}} = \frac{C_k W_k^3}{t_k^{'2}}$ and $E_k^{\text{GPU}} = \frac{G_k W_k^{'3}}{t_k^{'2}}$ denote the computation energy of CPU and GPU, respectively. Then, the necessary and sufficient condition for optimal workload allocation is

$$\frac{\partial E_k^{\text{CPU}}}{\partial W_k} = \frac{\partial E_k^{\text{GPU}}}{\partial W'_k}, \quad \forall k \in \mathcal{K}, \tag{12}$$

with $W_k + W'_k = W$.

Proof: See Appendix B. □

Furthermore, one can observe that the objective function in **P2** is a non-increasing function in ${t'}_k, \forall k \in \mathcal{K}$. Therefore, the optimality requires to maximize the computation time of each device, resulting in $t'^{*}_k = {T'}_k, \forall k \in \mathcal{K}$, which is independent of the workload partitioning. Using the result as well as Lemmas 1 and 2, the optimal computation RM policy is obtained as follows.

Proposition 1 (Optimal Computation RM): The optimal workload allocation is

(Optimal Workload Allocation)

$$W_k^* = \frac{\sqrt{G_k}W}{\sqrt{C_k}+\sqrt{G_k}}, \quad W'^{*}_k = \frac{\sqrt{C_k}W}{\sqrt{C_k}+\sqrt{G_k}}, \ k \in \mathcal{K}, \tag{13}$$

where C_k and G_k are computation coefficients for CPU and GPU, respectively. Moreover, the optimal speed scaling for CPU and GPU is

$$\text{(Optimal Speed Scaling)}\ f_k^* = \frac{W_k^*}{{T'}_k}, \ f'^{*}_k = \frac{W'^{*}_k}{{T'}_k}, \ k \in \mathcal{K}, \tag{14}$$

where T'_k is the allowed maximum computation time for device k.

Remark 2 (Energy-Efficient CPU-GPU Heterogeneous Computing): According to Proposition 1, more workload tends to be allocated to the processor with a smaller computation coefficient indicating a higher computation efficiency (see Remark 1), thereby reducing the devices' energy consumption. In addition, we emphasize on another fact that the proposed CPU-GPU heterogeneous computing scheme with policy in Proposition 1 is more energy-efficient than using the GPU (or CPU) only. Mathematical proof is presented in Appendix C.

B. Communication Resource Management

1) Problem Formulation: The total radio resources are managed by controlling bandwidth allocation and transmission time to minimize the sum transmission energy. The corresponding optimization problem can be formulated as

$$\begin{aligned} (\mathbf{P3}) \quad \min_{\{B_k, t_k\}} \quad & \sum_{k=1}^{K} \frac{B_k t_k N_0}{h_k^2}\left(2^{\frac{L}{B_k t_k}}-1\right) \\ \text{s.t.} \quad & \sum_{k=1}^{K} B_k = B, \quad B_k \geq 0, \ \forall k \in \mathcal{K}, \\ & 0 \leq t_k \leq T_k, \quad \forall k \in \mathcal{K}. \end{aligned}$$

The problem has a standard structure in the literature of energy-efficient communication (see e.g., [37]). It is convex and can be solved using a standard algorithm such as BCD. In the sequel, we present an alternative solution method yielding an equilibrium property that is useful for low-complexity policy computation faster than conventional methods.

2) Properties of Optimal Policies: As before, the objective of Problem **P3** is observed to be a non-increasing function in t_k. Therefore, it is optimal to maximize the transmission time of each device, resulting in $t_k^* = T_k, \forall k \in \mathcal{K}$. Next, to derive the optimal bandwidth allocation strategy, a necessary and sufficient condition is given as follows.

Lemma 3 (Energy-Bandwidth Rate Equilibrium): The *energy-bandwidth rate* as defined earlier is mathematically obtained as

$$\frac{\partial E_k^{\text{cmm}}}{\partial B_k} = \frac{t_k N_0}{h_k^2}\left(2^{\frac{L}{B_k t_k}} - \frac{L \ln 2}{B_k t_k} 2^{\frac{L}{B_k t_k}} - 1\right) < 0, \quad k \in \mathcal{K}. \tag{15}$$

The optimal bandwidth allocation equalizes the energy-bandwidth rates as

$$\frac{\partial E_1^{\text{cmm}}}{\partial B_1} = \frac{\partial E_2^{\text{cmm}}}{\partial B_2} = \cdots = \frac{\partial E_K^{\text{cmm}}}{\partial B_K} = -\nu^*, \tag{16}$$

where ν^* is a constant and $\sum_{k=1}^{K} B_k = B$.

Proof: See Appendix D. □

It can be observed from the above lemma that the increase of allocated bandwidth reduces the communication energy consumption of the device. Then, the optimal communication RM policy directly follows from the energy-bandwidth rate equilibrium as stated below.

Proposition 2 (Optimal Bandwidth Allocation): The optimal policy for bandwidth allocation is

$$B_k^* = \frac{L \ln 2}{T_k\left[1 + \mathcal{W}_0\!\left(\frac{h_k^2 \nu^* - T_k N_0}{T_k N_0 e}\right)\right]}, \quad k \in \mathcal{K}, \tag{17}$$

where T_k is the allowed maximum transmission time for device k; $\mathcal{W}_0(\cdot)$ is the Lambert W function (principal branch) and e is the Euler's number.

Proposition 2 suggests that B_k^* is a non-increasing function with respect to h_k^2. It means that more bandwidths should be allocated to devices with weaker channels for the benefit of sum communication energy reduction.

Obtaining the optimal bandwidths $\{B_k^*\}$ via (17) requires computing the optimal energy-bandwidth rate ν^* by solving the following equation:

$$\sum_{k=1}^{K} \frac{L \ln 2}{T_k\left[1 + \mathcal{W}_0\!\left(\frac{h_k^2 \nu^* - T_k N_0}{T_k N_0 e}\right)\right]} = B. \tag{18}$$

Due to the intractability of the Lambert function $\mathcal{W}_0$ and the unknown range of ν^*, the solution cannot be found efficiently using standard methods such as Newton-Raphson method and bi-section search. An alternative method, *derivative-free optimization* (DFO), which requires no derivative, has too high complexity. To address this issue, a fast algorithm for policy computation is designed below.

3) Optimal Policy Computation: Instead of solving ν^* and $\{B_k^*\}$ in a sequential order, the fast algorithm calculates the optimal values iteratively, comprising the following two phases.

- **Phase I**: (Bandwidth Optimization). Given the energy-bandwidth rate ν, compute the bandwidth allocation $\{B_k\}$ using (17). As ν is not optimal, the computed $\{B_k\}$ may not satisfy the bandwidth constraint in (7). Thus it is necessary to normalize each B_k as $\widetilde{B}_k = \frac{B}{\sum_{k=1}^{K} B_k} B_k$.

- **Phase II**: (Energy-Bandwidth Rate Updating). Calculate the respective energy-bandwidth rates $\nu_k = -\frac{\partial E_k^{\text{cmm}}}{\partial B_k}$, $\forall k \in \mathcal{K}$, by substituting $\{\widetilde{B}_k\}$ and $\{t_k = T_k\}$ into (15). Then, update the current energy-bandwidth rate ν using $\{\nu_k\}$ as elaborated in the sequel.

The two phases are iterated until convergence as indicated by the *energy-bandwidth rate equilibrium* in (16).

A key step in the algorithm is the updating rule for ν in Phase II. To derive the rule, a useful property is given as follows.

Lemma 4: Given $\nu_k(B_k) = -\frac{\partial E_k^{\text{cmm}}}{\partial B_k}$ in (15) and $B_k(\nu)$ in (17), $\forall k \in \mathcal{K}$, for the i-th iteration between Phase I and II, the following holds:

- If $\nu^{(i-1)} > \nu^*$, then 1) $\nu^* < \max\limits_k\{\nu_k^{(i)}\} < \nu^{(i-1)}$, and 2) $\text{sgn}\left(\sum_{k=1}^K B_k^{(i)} - B\right) = -1$;
- If $\nu^{(i-1)} < \nu^*$, then 1) $\nu^{(i-1)} < \min\limits_k\{\nu_k^{(i)}\} < \nu^*$, and 2) $\text{sgn}\left(\sum_{k=1}^K B_k^{(i)} - B\right) = 1$;

where $\nu_k^{(i)} = \nu_k(\widetilde{B}_k^{(i)})$ with $\widetilde{B}_k^{(i)} = \frac{B}{\sum_{k=1}^K B_k^{(i)}} B_k^{(i)}$ given $B_k^{(i)} = B_k(\nu^{(i-1)}), \forall k \in \mathcal{K}$.

Proof: See Appendix E. □

Essentially, by induction, the above Lemma 4 simply implies that if the initial point $\nu^{(0)}$ is greater than ν^*, and the updating rule adopted for ν is $\nu^{(i)} = \max\limits_k\{\nu_k^{(i)}\}$, the convergence of ν to the global optimal ν^* is guaranteed. This is also true for the case of $\nu^{(0)} < \nu^*$ by involving the updating rule as $\nu^{(i)} = \min\limits_k\{\nu_k^{(i)}\}$. It is further noted that the condition of $\nu^{(0)} > \nu^*$ or $\nu^{(0)} < \nu^*$ can be determined by calculating $\text{sgn}\left(\sum_{k=1}^K B_k^{(1)} - B\right)$, where $B_k^{(1)} = B_k(\nu^{(0)})$ follows from (17). In summary, we have the following updating rule for ν in Phase II:

$$\text{(Update rule)}\quad \nu^{(i)} = \begin{cases} \max\limits_k\{\nu_k^{(i)}\}, & \text{sgn}\left(\sum\limits_{k=1}^K B_k^{(1)} - B\right) = -1; \\ \min\limits_k\{\nu_k^{(i)}\}, & \text{otherwise}, \end{cases}$$

where $\{\nu_k^{(i)}\}$ and $\{B_k^{(1)}\}$ are specified in Lemma 4. Based on the above rule, the algorithm for optimal bandwidth allocation is summarized in Algorithm 1.

Remark 3 (Low-Complexity and Optimality): The complexity of our proposed Algorithm 1 is $O(\log\frac{1}{\varepsilon})$ with ε denoting the target accuracy. For comparison, the computation of the DFO method for solving ν^* and $\{B_k^*\}$ in a sequential order is $O(\frac{1}{\varepsilon^2})$, and that of the BCD algorithm for directly solving Problem **P3** is $O(\frac{K}{\varepsilon}\log\frac{1}{\varepsilon})$, which are much higher than that of Algorithm 1. Furthermore, the optimality is guaranteed by Algorithm 1 as indicated by Lemma 4.

Algorithm 1 Optimal Bandwidth Allocation

```
Input: initial value of $\nu$.
Output: optimal $\nu^*$ and $\{B_k^*\}$.
Calculate: indicator $y = \text{sgn}\left(\sum_{k=1}^K B_k(\nu) - B\right)$.
Repeat:
  • Calculate $\{B_k\}$ by substituting $\nu$ into Proposition 2;
  • Normalize $\widetilde{B}_k = \frac{B}{\sum_{k=1}^K B_k} B_k$, $k \in \mathcal{K}$;
  • Calculate $\{\nu_k\}$ by substituting $\{\widetilde{B}_k\}$ into (15), $k \in \mathcal{K}$;
  • Update $\nu = \frac{1-y}{2}\max\limits_k\{\nu_k\} + \frac{1+y}{2}\min\limits_k\{\nu_k\}$;
Until var$[\{\nu_k\}] < \varepsilon$ (to equalize the energy-bandwidth rates).
```

C. Joint C^2 Resource Management

1) Problem Formulation: Building on the preceding results, an energy-efficient C^{2}RM framework is designed by joint control of workload partitioning, C^2 time division, and bandwidth allocation to minimize sum energy. The optimization problem is formulated as

$$(\mathbf{P4})\quad \begin{aligned} \min_{\{W_k, t'_k, B_k, t_k\}} \quad & \sum_{k=1}^K \left[\frac{C_k W_k^3 + G_k {W'}_k^3}{{t'}_k^2} + \frac{B_k t_k N_0}{h_k^2}\left(2^{\frac{L}{B_k t_k}} - 1\right)\right] \\ \text{s.t.} \quad & W_k + W'_k = W,\ W_k \geq 0,\ W'_k \geq 0,\ \forall k \in \mathcal{K}, \\ & \sum_{k=1}^K B_k = B,\ B_k \geq 0,\ \forall k \in \mathcal{K}, \\ & t'_k + t_k \leq T, \quad t'_k \geq 0,\ t_k \geq 0,\ \forall k \in \mathcal{K}. \end{aligned}$$

Remark 4 (Feasibility Checking): The constraints on maximum frequency and transmit power are omitted to simplify the problem and the solution, because the current design focuses on the regime of energy-efficient system operation where such constraints are usually inactive. The current design can be modified to ensure the feasibility under such constraints by translating them into the requirements on the minimum computation time and bandwidth, which can be included as guidelines for deploying the current design. In particular, under the constraints: $f_k \leq f_{k,\max}$, $f'_k \leq f'_{k,\max}$, $P_k^{\text{TX}} \leq P_{k,\max}, \forall k \in \mathcal{K}$, the required bandwidth for **P4** is given as

$$B \geq \sum_{k=1}^K \frac{L}{\left(T - \frac{W}{f_{k,\max} + f'_{k,\max}}\right)\log_2\left(1 + \frac{P_{k,\max} h_k^2}{N_0}\right)}. \tag{19}$$

2) Properties of Optimal Policy: It is shown in the sequel that a set of equilibriums exist among devices in terms of energy rates with respect to different types of control variables when C^{2}RM is optimally energy-efficient. The insights facilitate designing low-complexity policy for solving Problem **P4**. First, by the same argument as in Subsections A and B, the latency constraint in Problem **P4** should be active for energy minimization: $t'_k + t_k = T, \forall k \in \mathcal{K}$. The optimal C^2 time division of T has the following property.

Lemma 5 (Energy-Time Rate Equilibrium): The *energy-time rates* as defined earlier can be mathematically obtained as

$$\xi'_k := \frac{\partial E_k^{\text{cmp}}}{\partial t'_k} = -\frac{2(C_k W_k^3 + G_k {W'}_k^3)}{{t'}_k^3}, \quad k \in \mathcal{K}, \tag{20}$$

and

$$\xi_k := \frac{\partial E_k^{\text{cmm}}}{\partial t_k} = \frac{B_k N_0}{h_k^2}\left(2^{\frac{L}{B_k t_k}} - \frac{L\ln 2}{B_k t_k} 2^{\frac{L}{B_k t_k}} - 1\right), \quad k \in \mathcal{K}. \tag{21}$$

The optimal C^2 time division for each device requires the energy-time rate equilibrium:

$$\frac{\partial E_k^{\text{cmp}}}{\partial t_k'} = \frac{\partial E_k^{\text{cmm}}}{\partial t_k}, \quad \forall k \in \mathcal{K}, \tag{22}$$

with $t_k' + t_k = T$.

The proof is similar to that for Lemma 2 and thus omitted for brevity. Next, one can observe that Problem **P4** integrates **P2** and **P3** by summing their objectives and combining their constraints. Therefore, **P4** is also convex and furthermore the results in Lemmas 2 and 3 hold for the current case. Then, combining Lemmas 1, 2, 3 and 5 yields the following main result.

Theorem 1 (Equilibrium Based C^{2}RM): The optimal C^{2}RM policy achieves the following equilibriums:

1) For each device, the optimal C^2 time division equalizes the energy-computation-time and energy-communication-time rates:

$$\frac{\partial E_k^{\text{cmp}}}{\partial t_k'} = \frac{\partial E_k^{\text{cmm}}}{\partial t_k}, \quad \forall k \in \mathcal{K}. \tag{23}$$

2) The optimal bandwidth allocation equalizes the energy-bandwidth rates:

$$\frac{\partial E_1^{\text{cmm}}}{\partial B_1} = \frac{\partial E_2^{\text{cmm}}}{\partial B_2} = \cdots = \frac{\partial E_K^{\text{cmm}}}{\partial B_K}. \tag{24}$$

3) The optimal workload allocation at each device equalizes the energy-workload rates:

$$\frac{\partial E_k^{\text{CPU}}}{\partial W_k} = \frac{\partial E_k^{\text{GPU}}}{\partial W_k'}, \quad \forall k \in \mathcal{K}. \tag{25}$$

4) The optimal speed scaling at each device equalizes the energy-speed rates:

$$\frac{\partial E_k^{\text{CPU}}}{\partial f_k} = \frac{\partial E_k^{\text{GPU}}}{\partial f_k'}, \quad \forall k \in \mathcal{K}. \tag{26}$$

Remark 5 (Equalizing C^2 Heterogeneity): In the process of synchronized updates, Theorem 1 suggests that energy-efficient C^{2}RM should equalize the heterogeneity in communication channels and computation efficiencies using the multi-dimensional control variables. First, the heterogeneity in multiuser channel states and CPU/GPU computation efficiencies at each device is equalized by bandwidth allocation and workload partitioning (with speed scaling) as reflected in the second and third (with fourth) equilibriums. Second, the heterogeneity in C^2 speeds is equalized by adjusting the C^2 time division according to the first equilibrium. It is worth mentioning that C^2 time division represents a *C^2 tradeoff* that more communication resources can compensate for the lack of computation resources and vice versa.

3) Optimal Policy Computation: Though the optimal solution for Problem **P4** can be obtained by finding the equilibriums in Theorem 1 numerically using the standard method of BCD, it has high complexity. To tackle this challenge, we design a more efficient algorithm as follows. First, Problem **P4** can be decomposed into one master problem and two sub-problems as follows:

- (Master Problem): The master problem is the optimization of C^2 time division. Denote $E^{\text{cmp}}(\{\widetilde{T}_k\})$ and $E^{\text{cmm}}(\{\widetilde{T}_k\})$ as the computation and communication energy, which are functions of the allowed maximum communication time $\{\widetilde{T}_k\}$ given optimized resource management strategies output by sub-problem 1 and sub-problem 2. Then, the master problem is cast as

$$(\mathbf{MP}) \quad \min_{\{\widetilde{T}_k\}} \; E^{\text{cmp}}(\{\widetilde{T}_k\}) + E^{\text{cmm}}(\{\widetilde{T}_k\})$$
$$\text{s.t. } 0 < \widetilde{T}_k < T, \quad \forall k \in \mathcal{K}.$$

- (Two Sub-problems):
  - (Computation RM): Problem **P1** with $\{T_k' = T - \widetilde{T}_k\}$;
  - (Communication RM): Problem **P3** with $\{T_k = \widetilde{T}_k\}$.

As Problem **P4** is convex, the optimal solutions can be obtained via iterations between the master problem and two sub-problems. Each iteration comprises two steps: 1) given $\{\widetilde{T}_k\}$, compute the optimal RM strategies by solving two sub-problems; 2) given allocated bandwidths and partitioned workloads, calculate the sub-gradient of the master problem and apply it to updating $\{\widetilde{T}_k\}$ via the gradient descent method. The two steps are iterated until convergence.

Note that the solution of one sub-problem, namely Problem **P1**, can be calculated directly via (13) and (14) while numerically calculation is required for solving the other sub-problem, namely Problem **P3**. Given the efficient Algorithm 1 developed for computing $\{B_k^*\}$, in the sequel, we aim at further improving the efficiency by proposing a novel initialization method of ν. To this end, another useful property is introduced in the following lemma.

Lemma 6: The optimal ν^* under the arbitrary given C^2 time division, namely $T_k' + T_k = T$ with $t_k' \leq T_k'$ and $t_k \leq T_k$, $\forall k \in \mathcal{K}$, can be calculated as

$$\nu^* = -\frac{1}{B}\sum_{k=1}^{K} T_k \frac{\partial E_k^{\text{cmm}}}{\partial t_k}\bigg|_{T_k} = -\frac{1}{B}\sum_{k=1}^{K} T_k \xi_k(T_k), \tag{27}$$

where $\xi_k(T_k) = \frac{\partial E_k^{\text{cmm}}}{\partial t_k}\big|_{T_k}$ is the energy-communication time rate under current time division.

The above result can be proved by noting the relation $T_k \frac{\partial E_k^{\text{cmm}}}{\partial t_k}\big|_{T_k} = B_k^* \frac{\partial E_k^{\text{cmm}}}{\partial B_k}\big|_{B_k^*} = -\nu^* B_k^*$ given the bandwidth constraint, namely $\sum_{k=1}^{K} B_k^* = B$. As the values of $\{\xi_k(T_k)\}$

are not attainable, ν^* can not be solved directly. Nevertheless, we can get a good initial point for ν by taking advantage of (27). To be specific, due to the fact that $\xi_k'(T_k')$ and $\xi_k(T_k)$ converge with iterations and $\xi_k'(T_k')$ can be calculated by solving **P1** in closed-form, we propose to initialize ν as

$$\nu_0 := -\frac{1}{B}\sum_{k=1}^{K} T_k \xi_k'(T_k'), \tag{28}$$

Such initialization provides an increasingly better initial point that is closer to the optimal solution as the outer iteration proceeds. The algorithm for computing the optimal C^{2}RM policy is summarized in Algorithm 2.

Algorithm 2 Optimal C^{2}RM

```
Input: initial values of $\{\widetilde{T}_k\}$.
Output: optimal $\{B_k^*\}$, $\{{t'}_k^* = T - \widetilde{T}_k^*\}$, $\{t_k^* = \widetilde{T}_k^*\}$.
Repeat:
  • Solve the two subproblems with $\{T_k = \widetilde{T}_k\}$ and $\{T_k' = T - \widetilde{T}_k\}$ as input:
      ▷ Solve Problem $(\mathbf{P1})$:
          - calculate $\{W_k\}$ and $\{W_k'\}$ using (13);
          - obtain $\{\xi_k'\}$ by substituting $\{t_k' = T_k'\}$ and $\{W_k, W_k'\}$ into (20);
      ▷ Solve Problem $(\mathbf{P3})$:
          - calculate $\{B_k\}$ and solve $\nu$ using Algorithm 1 by involving the initialization method (28);
          - obtain $\{\xi_k\}$ by substituting $\{B_k\}$ and $\{t_k = T_k\}$ into (21);
  • Update the time division: $\widetilde{T}_k = \widetilde{T}_k - \eta(\xi_k - \xi_k')$, $\forall k \in \mathcal{K}$;
Until $\|\boldsymbol{\xi}' - \boldsymbol{\xi}\|^2 \le \varepsilon$ (to equalize the energy-time rates), where $\boldsymbol{\xi}' \triangleq [\xi_1', \cdots, \xi_K']^\top$ and $\boldsymbol{\xi} \triangleq [\xi_1, \cdots, \xi_K]^\top$.
```

Remark 6 (Low-Complexity Algorithm): The proposed Algorithm 2 is of low-complexity, which has complexity up to $O(K\log^2\frac{1}{\varepsilon})$ as compared to that of the conventional BCD method for solving **P4**, which has complexity $O(\frac{K}{\varepsilon}\log\frac{1}{\varepsilon})$.

D. Discussion on Energy-Learning Tradeoff

In the literature, the model convergence of federated learning is characterized by either the averaged gradient norm (see e.g., [35]) or the loss function (see e.g., [23]), as a monotone decreasing function of the number of required rounds, N, and participating devices K. For example, it is reported in [35] that with a properly chosen learning rate (step size), the averaged gradient norm over rounds is shown to be proportional to $1/\sqrt{N}$ and $1/\sqrt{K}$. In existing work, the per-round latency, T, is assumed constant and thereby can be omitted in the convergence analysis. By considering T or N as a function of C^2 sum energy, denoted as E_Σ, we can discuss a learning-energy tradeoff as follows.

*Definition 1 (*Learning Latency*):* The learning latency is defined as $T_{\text{total}} = N \times T$ (in second).

1) Fixed Number of Participating Devices, K: This implies a fixed distributed dataset and thus fixed N. On the other hand, it can be inferred from Lemma 5 that T is a monotone decreasing function of E_Σ. Thus, the learning latency can be written as $T_{\text{total}} = N \times T(E_\Sigma)$, which establishes the tradeoff between T_{total} and E_Σ. Specifically, reducing E_Σ can result in increased learning latency, and vice versa. Based on Lemma 5, one can see that the reduction in the learning latency leads to the increment in sum energy at a rate *faster than linear*.

2) Fixed Per-Round Latency, T: For this case, an energy-learning tradeoff also exists. To be specific, it can be easily shown that the number of devices that can satisfy the per-round latency constraint is a monotone increasing function of E_Σ. Thereby, reducing E_Σ can result in a reduced number of devices participating in learning. This gives rise to a larger required number of rounds for convergence due to the shrinking of distributed dataset. Thus, the learning latency can be written as $T_{\text{total}} = N(E_\Sigma) \times T$ with $N(E_\Sigma)$ being a monotone decreasing function. Then the energy-learning tradeoff is one that shortening learning latency needs more energy and vice versa. Particularly, the intermediate parameter, namely the number of participating devices, controls the said tradeoff in this case. This further motivates us to explore C^2 aware device scheduling in the following section.

In general, the functions $T(E_\Sigma)$ in the first case and $N(E_\Sigma)$ in the second case have no closed form. The analysis of their properties (e.g., scaling laws) is an interesting topic but outside the scope of this work.

IV. C^2 Aware Scheduling

When there are many devices providing more than sufficient data, it is desirable from the energy-efficiency perspective to select only a subset of devices for participating in model training. In this section, the scheduler design is presented and the effect of scheduling on model convergence is quantified.

A. Scheduler Design

The mathematical formulation of the design is similar to **P4** by modifying the objective as $\sum_{k=1}^{K}\rho_k\Big[\frac{C_k W_k^3 + G_k {W'}_k^3}{{t'}_k^2} + \frac{B_k t_k N_0}{h_k^2}\Big(2^{\frac{L}{B_k t_k}} - 1\Big)\Big]$ under the additional constraints, namely $\sum_{k=1}^{K}\rho_k = M,\ \rho_k \in \{0,1\},\ \forall k \in \mathcal{K}$, where $\{\rho_k\}$ are the indicator variables for selecting devices. This mixed integer non-linear problem is NP-hard and classical optimal algorithms (e.g., branch and bound) have too high complexity to be practical when K is large. To tackle this challenge, we propose a novel C^2 aware scheduler design of low-complexity to minimize sum energy.

We consider the problem of selecting M out of K devices for FEEL. The data distribution over devices is assumed i.i.d. Then with M fixed, device scheduling has no effect on the number of rounds N required for learning (whose dependence on M is studied in Section B). This allows scheduling decisions to be based on the C^2 states of devices, giving the name of C^2 aware scheduling. The key element of the scheduler is a scheduling metric designed as follows. Based on the preceding analysis, it is desirable to select devices with good channels or/and good computation efficiencies for energy reduction. To identify such devices, we propose to first perform equal bandwidth allocation over all devices (i.e., $B_k = \bar{B} = B/K,\ \forall k \in \mathcal{K}$) and then evaluate the resulting energy consumption of each device. Note that equal bandwidth allocation enables the high energy consumption of a device to indicate: 1) a poor channel, or 2) a low computation efficiency, or 3) both. Therefore, the devices' energy consumption with equal bandwidth allocation is a suitable scheduling metric given as:

$$E_k = \frac{\bar{B} t_k^* N_0}{h_k^2}\left(2^{\frac{L}{\bar{B} t_k^*}} - 1\right) + \frac{a_k W^3}{(T - t_k^*)^2}, \quad \forall k \in \mathcal{K}, \tag{29}$$

where $a_k \triangleq \frac{C_k G_k}{(\sqrt{C_k}+\sqrt{G_k})^2}, \forall k \in \mathcal{K}$. To compute its value, the optimal transmission time t_k^* is needed given $B_k = \bar{B} \triangleq B/K$. Based on the energy-time rate equilibrium developed in Lemma 5, t_k^* solves the equation below:

$$f(t_k^*) = \frac{\bar{B} N_0}{h_k^2}\left(2^{\frac{L}{\bar{B} t_k^*}} - \frac{L \ln 2}{\bar{B} t_k^*} 2^{\frac{L}{\bar{B} t_k^*}} - 1\right) + \frac{2 a_k W^3}{(T - t_k^*)^3} = 0. \tag{30}$$

A closed-form expression for t_k^* is hard to derive but can be numerically computed by a bi-section search since $f(t_k^*)$ is a monotonically increasing function. Given the values of $\{E_k\}$, the C^2 aware scheduler selects M devices with the smallest values.[4] The scheduling scheme is summarized in Algorithm 3.

Algorithm 3 C^2 Aware Scheduling

```
Initialization: $\forall k \in \mathcal{K},\ \rho_k = 0,\ \bar{B} = B/K$.
Output: Subset $\mathcal{M} = \{k \in \mathcal{K} \mid \rho_k = 1\}$.
  • Solve for the optimal time divisions $\{t_k^*\}$ using (30) and a bi-section search;
  • Calculate the scheduling metrics $\{E_k\}$ by substituting $\{t_k^*\}$ into (29);
  • Select $M$ devices with the smallest $E_k$ and set their $\rho_k$=1.
```

B. Effect of Scheduling on Convergence

In this subsection, we quantify the effect of C^2 aware scheduling (without considering data importance) on the convergence rate of FEEL (in round) due to the reduced size of selected dataset for model updating. For tractable analysis, we follow the standard assumptions on the loss function as made in the literature (see e.g., [38]).

Assumption 1: It is assumed that the loss functions has the following two properties:

- (Convexity and Smoothness). All functions $\{F_k\}$ are convex and β-smooth, that is, for all $\mathbf{u}, \mathbf{v}$, we have

$$\begin{aligned} F_k(\mathbf{u}) &\geq F_k(\mathbf{v}) + \langle \nabla F_k(\mathbf{v}), \mathbf{u} - \mathbf{v} \rangle, \\ F_k(\mathbf{u}) &\leq F_k(\mathbf{v}) + \langle \nabla F_k(\mathbf{v}), \mathbf{u} - \mathbf{v} \rangle + \frac{\beta}{2}\|\mathbf{u} - \mathbf{v}\|^2. \end{aligned} \tag{31}$$

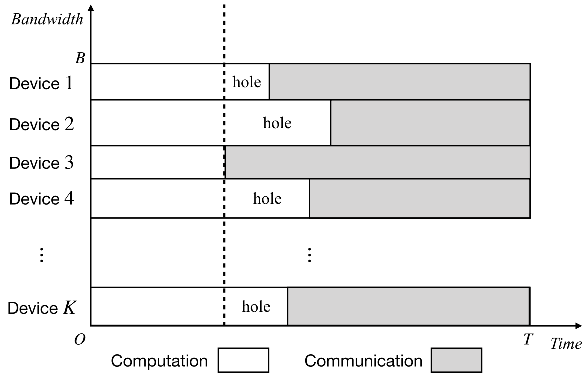


Fig. 2. Spectrum holes exist due to heterogeneous C^2 time division.

- (Variance Bound). Stochastic gradients $\{\nabla F_k(\mathbf{w})\}$ are unbiased and variance bounded by σ^2, that is,

$$\begin{aligned} &\mathbb{E}[\nabla F_k(\mathbf{w})] = \nabla F(\mathbf{w}), \\ &\mathbb{E}[\|\nabla F_k(\mathbf{w}) - \nabla F(\mathbf{w})\|^2] \leq \sigma^2, \end{aligned} \tag{32}$$

  where $\nabla F_k(\mathbf{w})$ and $\nabla F(\mathbf{w})$ denote the gradients of a local loss function and the global loss function, respectively, and the expectations are taken over all devices.

Under the above assumptions, the upper bound on the convergence rate of FEEL algorithm with device scheduling is given in the following theorem.

Theorem 2 (Convergence Rate with Scheduling): Given the learning rate as $\eta = \frac{1}{\sqrt{N}}$, the convergence rate of the FEEL algorithm with device-scheduling can be upper-bounded as

$$\begin{aligned} &\frac{1}{N}\sum_{i=0}^{N-1} \mathbb{E}_{\mathcal{M}_i}\left[F(\mathbf{w}^{(i)}) - F(\mathbf{w}^*)\right] \\ &\qquad \leq \frac{1}{\sqrt{N}}\left[\|\mathbf{w}^{(0)} - \mathbf{w}^*\|^2 + \frac{\sigma^2(K-M)}{(K-1)M}\right]. \end{aligned} \tag{33}$$

where N denotes the number of communication rounds.

For brevity, the detailed proof is omitted here and presented in the extended version [39]. Compared with the existing results without scheduling (see e.g., [40]), the last term in (33) is new that characterizes the effect of scheduling on learning. One can observe from the term that increasing the number of scheduled devices M leads to the vanishment of the biased term at a rate of $O(\frac{1}{M})$, giving rise to a faster convergence rate, however, at the cost of more sum energy consumption scales *faster than linearly* with M.

V. Extension and Discussion

A. Greedy Spectrum Sharing

In the preceding sections, the bandwidths are allocated at the beginning of each round and then fixed throughout the round. However, as mentioned, the heterogeneity in the computation durations of devices may result in spectrum holes as illustrated in Fig. 2. Scavenging them by spectrum sharing among devices can reduce sum energy, which is the theme of this section. While optimizing the spectrum sharing is intractable, we design a practical scheme based on the greedy principle. The scheme allocates the spectrum hole upon its arrival to one of available devices for transmission (in addition to the

[4] Given M scheduled devices, Algorithm 2 is applied to calculating optimal joint C^{2}RM policy for them.

pre-allocated bandwidth) so as to reduce its energy consumption, called *greedy spectrum sharing*. To design the scheme, we divide the round into multiple time slots. Furthermore, let $\mathcal{S}$ and $\mathcal{S}_\ell$ denote the sets of scheduled devices and a subset of devices that have completed computation at the beginning of a particular time slot, respectively. Then, the resulting spectrum holes have the total bandwidth of $B - \sum_{k\in\mathcal{S}_\ell} B_k^*$ with B_k^* denoting the pre-allocated bandwidth to device k, which is determined at the beginning of each round. Intuitively, all the unoccupied bandwidths should be allocated to the device $k \in \mathcal{S}_\ell$ that has the largest energy reduction with respect to bandwidth growth. Mathematically, this selection metric is to select the device, denoted as k^*, with the minimum energy-bandwidth rate at B_k^* provided $\frac{\partial E_k^{\text{cmm}}}{\partial B_k} < 0, \forall k \in \mathcal{K}$, that is

$$k^* = \arg\min_{k\in\mathcal{S}_\ell} \left.\frac{\partial E_k^{\text{cmm}}}{\partial B_k}\right|_{B_k^*}. \tag{34}$$

However, according to Lemma 3, the energy-bandwidth rates of all scheduled devices are equalized at equilibrium, making this criterion ineffective. To address this issue, we design an alternative metric that can also effectively reduce sum energy. To this end, we introduce the following notion.

Definition 2 (Acceleration Rate): The *acceleration rate* is defined as the partial derivative of the energy-bandwidth rate:

$$\phi_k(B_k) := \frac{\partial^2 E_k^{\text{cmm}}}{\partial B_k^2} = \frac{2^{\frac{L}{B_k t_k}} L^2 N_0 (\ln 2)^2}{B_k^3 t_k h_k^2} > 0,\ \forall k \in \mathcal{K}. \tag{35}$$

A device with a small acceleration rate implies a lower energy-bandwidth rate upon being allocated with extra bandwidths, leading to more energy reduction. Therefore, we propose to adopt the criterion of minimum acceleration rate at $\{B_k = B_k^*\}$, namely $\{\phi_k(B_k^*)\}$, as follows

$$\text{(Device Selection)} \quad k^* = \arg\min_{k\in\mathcal{S}_\ell} \phi_k(B_k^*). \tag{36}$$

Based on the above criterion, the scheme is summarized in Algorithm 4.

Algorithm 4 Greedy Spectrum Sharing

```
Initialization: Apply Algorithm 3 and 2 in sequential order to obtain $\{\rho_k^*, B_k^*\}$.
Denote $\Delta t$ as the time slot duration and let $t_{\text{count}} = 0$.
For the subset of devices $\mathcal{S} = \{k \in \mathcal{K} \mid \rho_k = 1\}$:
While $t_{\text{count}} < T$:
  • Denote $\mathcal{S}_\ell$ as the set of devices that have completed local computation at time $t_{\text{count}}$;
  • For $k \notin \mathcal{S}_\ell$, no bandwidth will be occupied by them;
  • For $k \in \mathcal{S}_\ell$, each device will first be allocated with bandwidth $B_k^*$;
      – Calculate $\phi_k$ for each $k \in \mathcal{S}_\ell$ using (35);
      – Select the device via (36) and allocate all the accessable spectrum $B - \sum_{k\in\mathcal{S}_\ell} B_k^*$ to it;
  • $t_{\text{count}} = t_{\text{count}} + \Delta t$.
```

B. Extension to Frequency-Selective Channels

The joint C^{2}RM and device scheduling algorithms in the preceding sections are designed assuming frequency non-selective channels. In this subsection, we discuss the deployment of the algorithms in the case of frequency selective channels in an *orthogonal frequency division multiple access* (OFDMA) system. Essentially, the optimization of the joint design needs to account for one more dimension, i.e., channel variation in frequency for each device. One practical approach for applying the current design is to consider the average channel gain for each device as in [41], [42]. To be specific, for each device, say device k, divide its frequency-selective channel into $|\mathcal{N}_k|$ frequency non-selective sub-channels with $\mathcal{N}_k$ being the sub-channel index set and $h_k^{(i)}$ denoting the channel gain of the i-th sub-channel. Then its average channel gain is given by

$$\bar{h}_k = \frac{\sqrt{\sum_{i\in\mathcal{N}_k} \left(h_k^{(i)}\right)^2}}{|\mathcal{N}_k|}, \quad \forall k \in \mathcal{K}. \tag{37}$$

By exploiting the average channel gain $\bar{h}_k$ as a surrogate for h_k of device k defined for the frequency non-selective counterpart in preceding sections, all the resource allocation strategies and algorithms developed for the frequency non-selective case can be applied to the current scenario.

VI. Experimental Results

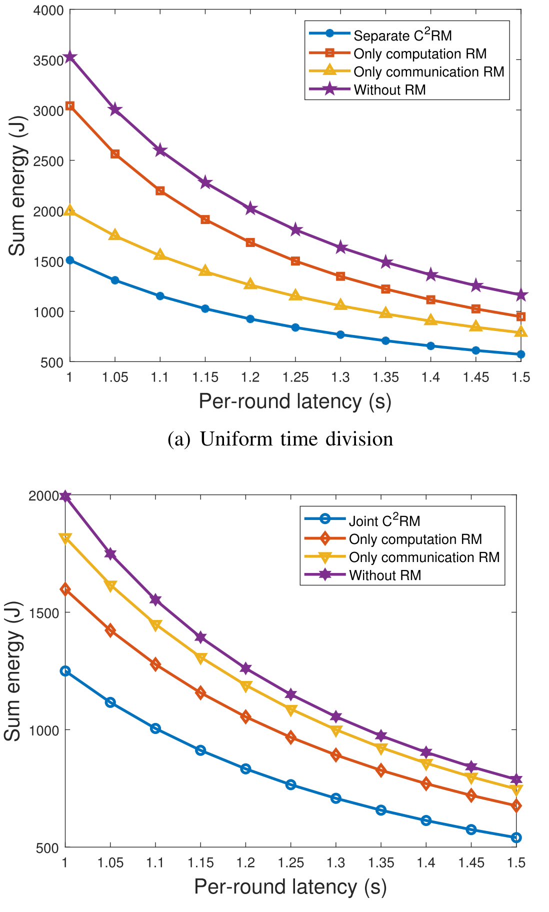
A. Experiment Setup

The simulation settings are as follows unless specified otherwise. In the FEEL system, there are $K = 50$ devices and each of them is capable of CPU-GPU heterogeneous computing. The devices' CPU and GPU coefficients, $\{C_k\}$ and $\{G_k\}$, are uniformly selected from the the set $\{0.020, 0.021, \cdots, 0.040\}$ and $\{0.001, 0.002, \cdots, 0.010\}$ [in Watt$\cdot$(MFLOPs/s)$^{-3}$], respectively.[5] Consider an FDMA system with the uplink bandwidth $B = 5$ MHz. The channel gains $\{h_k\}$ are modeled as i.i.d. Rayleigh fading with average path loss set as 10^{-3}. The noise variance is $N_0 = 10^{-9}$ W/Hz. The classification task aims at classifying handwritten digits from the well-known MNIST dataset. The classifier model is implemented using a 6-layer *convolutional neural network* (CNN) which consists of two 5×5 convolution layers with ReLU activation, each followed by 2×2 max pooling, a fully connected layer with 50 units and ReLU activation, and a final softmax output layer. The total number of parameters is $21,840$ and the computation workload is $W = 9.75$ MFLOPs. Furthermore, we suppose that each parameter of the training model gradient is quantized into 16 bits, and as a result, the transmission overhead is $L = 3.49 \times 10^5$ bits.

B. C^2 Resource Management

1) Energy-Efficient RM: The performance of the proposed computation RM, communication RM and joint C^{2}RM policies are evaluated by simulations. To be specific, we consider

[5] The range of $\{C_k\}$ and $\{G_k\}$ are set according to the fact that local computation energy consumption per round should be within the range based on the real test, e.g., the test on the Samsung Exynos 5422 SoC [14].



(a) Uniform time division

(b) Optimal time division

Fig. 3. Comparison of the energy efficiencies of C^{2}RM, only computation RM, only communication RM, and without RM schemes, with the (a) uniform time division and (b) optimal time divisions, respectively.

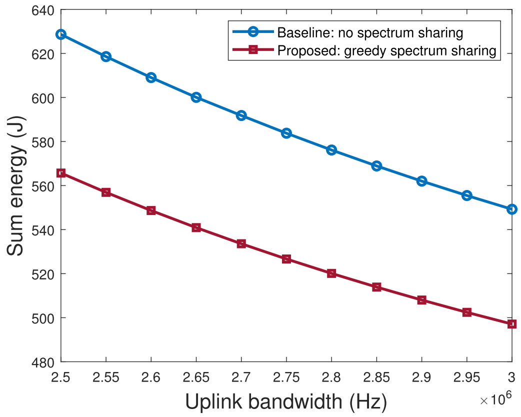


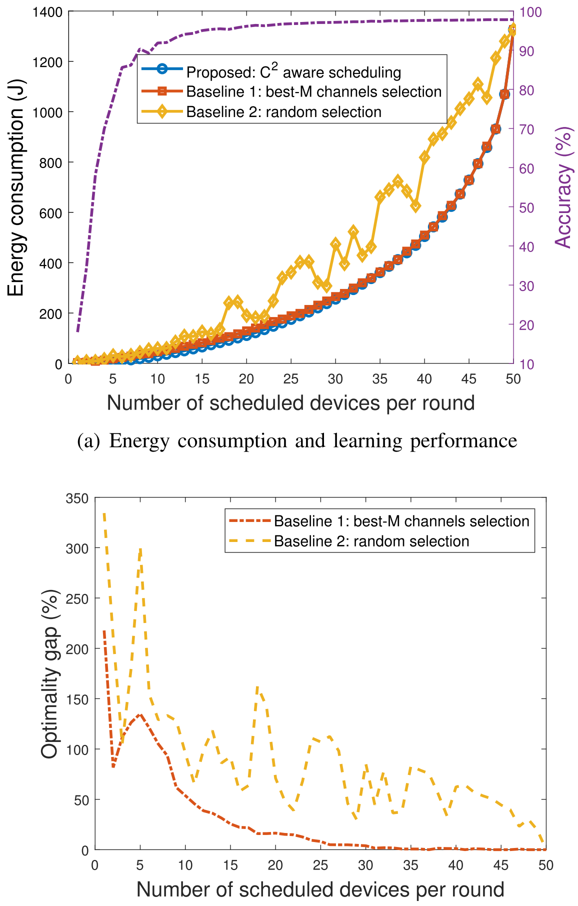
Fig. 4. Comparison between joint C^{2}RM with and without greedy spectrum sharing. The lines show sum energy vs. uplink bandwidth with fixed one round time $T = 1$ s.

two settings: a) uniform time division [see Fig. 3(a)]; and b) optimal time divisions [see Fig. 3(b)]. Under each setting, we consider four schemes: 1) C^{2}RM, 2) only computation RM, 3) only communication RM, and 4) without RM. In particular, to distinguish our proposed optimal C^{2}RM policy, we name the policy with C^{2}RM under uniform time division setting as "separate C^{2}RM" while the proposed optimal one is called "joint C^{2}RM". The curves of the sum energy versus the per-round latency T are illustrated in Fig. 3. Serveral observations can be made. First, the sum energy reduces as T grows for all cases. This coincides with the results in Lemma 5 that the energy consumption is a monotonically decreasing function in computation and communication time. Second, for either setting a) or b), it can be found that the two schemes 2) and 3), namely only computation RM and only communication RM, outperform the scheme without RM but underperform the C^{2}RM. For example, under setting a), they reduce the sum energy of the policy without RM by 13.8% and 43.5%, respectively, for per-round latency equal to 1.0 s. Meanwhile, the separate C^{2}RM scheme reduces the sum energy of the policy without RM by 57.3%. These results demonstrate the effectiveness of our proposed workload and bandwidth allocation policies for C^{2}RM. Third, comparing the results in setting a) with b), we find that the strategy of C^2 time division plays a significant role in energy efficiency. For example, the four schemes 1)-4) in b) with optimal time divisions reduce the sum energy of those in a) by 17.2%, 47.5%, 9.8%, and 43.5%, respectively, for per-round latency equal to 1.0 s. This coincides with Remark 4 that the optimal C^2 time division achieves the best energy efficiency by balancing C^2 heterogeneity. Fourth, our proposed joint C^{2}RM policy outperforms all other schemes, showing its effectiveness. For example, it reduces the sum energy of the schemes 1)-4) in a) by 17.2%, 59.0%, 37.4%, and 64.6%, as well as the schemes 2)-4) in b) by 21.9%, 31.3%, and 37.4%, respectively, for per-round latency equal to 1.0 s.

2) Greedy Spectrum Sharing: The performance of the proposed greedy spectrum sharing algorithm in Algorithm 4 is benchmarked against the optimal C^{2}RM without spectrum sharing in Algorithm 2. Note that the number of devices are set to be $K = 20$ with high heterogeneity regarding their computation efficiencies and channels. The curves of sum energy versus the bandwidth B are plotted in Fig. 4. Two observations can be made. First, the sum energy reduces as B grows as more bandwidths can be traded for lower transmission power. Second, it can be found that the proposed greedy spectrum sharing policy improves the energy efficiency of the C^{2}RM without spectrum sharing by scavenging unused radio resources. For example, it reduces sum energy of the baseline scheme, namely optimal C^{2}RM without spectrum sharing by 10.6% for uplink bandwidth equal to 2.5 MHz.

C. Device Scheduling

Consider the scenario that the edge server schedules a subset of devices for learning. The number of rounds is fixed as 10 with constrained latency $T = 1$ s for each round. The performance of the proposed C^2-aware scheduling is benchmarked against two baselines: 1) best-M channels selection and 2) the random selection. The effects of the number of scheduled devices M on the average learning accuracy of the FEEL algorithm and sum energy consumption are shown in Fig. 5(a). Several observations can be made as follows. First, the average learning accuracy is an increasing function of M as it increases the training data per round. Second, it can be observed that the increase on M leads to the



(a) Energy consumption and learning performance

(b) Optimality gap

Fig. 5. (a) The comparison of the energy efficiencies between C^2 aware scheduling and two baselines is shown by solid lines. The relationship between test accuracy and the number of scheduled devices per round is illustrated by the dash line. (b) Comparison of optimality gap of best-M channels selection (baseline 1) and random selection (baseline 2) w.r.t. C^2 aware scheduling (proposed). The optimality gaps for the baseline is defined as $\frac{E_{\text{baseline}} - E_{\text{proposed}}}{E_{\text{proposed}}} \times 100\%$, where E_{proposed} and E_{baseline} denote the sum energy consumption with the solution from C^2 aware scheduling and the baseline.

growth of sum energy at a rate faster than linear, which agrees with the preceding discussion. Furthermore, one can observe that the proposed scheduling scheme outperforms the baseline which randomly selects M devices, e.g., achieving 45.1% sum energy reduction for $M = 35$. To present the results more clearly, we define the optimality gap for the baseline (either best-M channels selection or random selection) as $\frac{E_{\text{baseline}} - E_{\text{proposed}}}{E_{\text{proposed}}} \times 100\%$, where E_{proposed} and E_{baseline} denote the sum energy consumption with the solution from C^2 aware scheduling and the baseline, respectively. It can be observed from Fig. 5(b) that the proposed C^2 aware scheduling also outperforms the baseline which selects M devices with best channels, e.g., the resultant optimality gap is up to 135% for $M = 5$. It is also worth mentioning that when the number of participants becomes large, the communication energy becomes the bottleneck due to the constrained total bandwidth. Therefore, both the proposed C^2 aware scheduling and best-M channels method tend to select the same subset of devices in this scenario.

VII. Concluding Remarks

In this paper, we have proposed a set of schemes for joint C^{2}RM to enhance the energy-efficiency of a FEEL system using CPU-GPU heterogeneous computing. This work paves the way for followup works to address several interesting issues, including scheduling given non-i.i.d. data distributions, optimal joint C^{2}RM for fast fading channels, and the consideration of state-of-the-art techniques ranging from OFDMA to NOMA to massive MIMO.

Appendix A
Proof of Lemma 1

Consider two cases for the second constraint in Problem **P2**. First, we consider the case that $\frac{W_k}{f_k} \geq \frac{W_k'}{f_k'}, k \in \mathcal{K}$. From equation (4), we know that $t'_k = \max\{\frac{W_k}{f_k}, \frac{W_k'}{f_k'}\} = \frac{W_k}{f_k}$. Therefore, we have $f_k = \frac{W_k}{t'_k}$ and $f_k' \geq \frac{W_k'}{t'_k}$. It follows that the computation energy consumption is

$$E_k^{\text{cmp}} = C_k W_k f_k^2 + G_k W_k' f_k'^2 \geq C_k W_k \frac{W_k^2}{t'^2_k} + G_k W_k' \frac{W'^2_k}{t'^2_k}. \tag{38}$$

The equality holds if and only if $f_k' = \frac{W_k'}{t'_k}$, meaning $\frac{W_k}{f_k} = \frac{W_k'}{f_k'}$. It is the necessary and sufficient condition for achieving minimum computation energy.

Then, it is also true for the case $\frac{W_k}{f_k} < \frac{W_k'}{f_k'}$.

Appendix B
Proof of Lemma 2

The computation energy of device k can be expressed as $E_k^{\text{cmp}} = E_k^{\text{CPU}}(W_k, t'_k) + E_k^{\text{GPU}}(W_k', t'_k)$, $\forall k \in \mathcal{K}$. The partial Lagrangian function is defined as

$$\begin{aligned} &\mathcal{L}(\{W_k\}, \{W_k'\}, \{\gamma_k\}, \{\lambda_k\}, \{\theta_k\}) \\ &= \sum_{k=1}^{K} [E_k^{\text{CPU}}(W_k, t'_k) + E_k^{\text{GPU}}(W_k', t'_k) \\ &\quad + \gamma_k(W_k + W'_k - W) - \lambda_k W_k - \theta_k W'_k], \end{aligned} \tag{39}$$

where $\{\lambda_k \geq 0\}$ and $\{\theta_k \geq 0\}$ are the Lagrange multipliers. Then we have the conditions:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial W_k} &= \frac{\partial E_k^{\text{CPU}}}{\partial W_k} + \gamma_k - \lambda_k = 0, \quad \forall k \in \mathcal{K}, \\ \frac{\partial \mathcal{L}}{\partial W'_k} &= \frac{\partial E_k^{\text{GPU}}}{\partial W'_k} + \gamma_k - \theta_k = 0 \quad \forall k \in \mathcal{K}. \end{aligned} \tag{40}$$

The complementary slackness conditions give that

$$\lambda_k W_k = 0, \quad \theta_k W'_k = 0, \quad \forall k \in \mathcal{K}. \tag{41}$$

Since at least one processor should be active. Assume $W'_k > 0$ and then we have $\theta_k = 0$ from the second condition in (41). Since the multiplier λ_k is required to be $\lambda_k \geq 0$, we know

$$\lambda_k = \frac{\partial E_k^{\text{CPU}}}{\partial W_k} - \frac{\partial E_k^{\text{GPU}}}{\partial W'_k} \geq 0, \quad \forall k \in \mathcal{K}. \tag{42}$$

Substituting results (11) of Lemma 2 into (42), we have $W_k \geq \sqrt{\frac{G_k}{C_k}} W'_k > 0$, $\forall k \in \mathcal{K}$. Accordingly, we have

$W_k > 0$ so that $\lambda_k = 0$. Therefore, it is optimal for both processors to be active with the energy-workload rate equilibrium (12) as stated in Lemma 2.

Appendix C
Proof of the Claim in Remark 2

We prove the fact that the proposed CPU-GPU heterogeneous computing scheme with policy in Proposition 1 is more energy-efficient than just using GPU to complete workload W within the allowed computation time T_k'. Based on (13) and (14), the optimal computation energy consumption of CPU-GPU heterogeneous computing is

$$\begin{aligned} E_{k,\text{CPU-GPU}}^{\text{cmp}} &= C_k W_k^* {f_k^*}^2 + G_k {W_k'}^* {f'}_k^{*2} \\ &= \frac{C_k G_k W^3}{\left(\sqrt{C_k}+\sqrt{G_k}\right)^2 {T'}_k^2}. \end{aligned} \tag{43}$$

As for the baseline that the GPU takes the total workload W, it is not hard to derive the optimal speed scaling rule as ${f'}_k^{*} = \frac{W}{T_k'}$ and the optimal computation energy is

$$E_{k,\text{GPU}}^{\text{cmp}} = G_k W {f'}_k^2 = \frac{G_k W^3}{{T'}_k^2}. \tag{44}$$

Then, we can derive the difference between them as follows:

$$E_{k,\text{CPU-GPU}}^{\text{cmp}} - E_{k,\text{GPU}}^{\text{cmp}} = -\frac{(\sqrt{G_k}W)^3 \left(2\sqrt{C_k}+\sqrt{G_k}\right)}{\left(\sqrt{C_k}+\sqrt{G_k}\right)^2 {T'}_k^2} < 0. \tag{45}$$

Therefore, we can conclude that $E_{k,\text{CPU-GPU}}^{\text{cmp}} < E_{k,\text{GPU}}^{\text{cmp}}$, which means the considered CPU-GPU heterogeneous computing is a more energy-efficient scheme.

Appendix D
Proof of Lemma 3

Substituting $t_k^* = T_k$ into (9), the communication energy of device k can thus be expressed as the function of allocated bandwidths, i.e., $E_k^{\text{cmm}}(B_k)$. By introducing Lagrange multipliers $\{\mu_k^*\}$ for the inequality constraints $\{B_k \geq 0\}$ and a scalar multiplier ν^* for the equality constraint $\sum_{k=1}^K B_k = B$, the Lagrangian function is defined as

$$\begin{aligned} &\mathcal{L}(\{B_k\}, \{\mu_k\}, \nu) \\ &\quad = \sum_{k=1}^{K} \left[E_k^{\text{cmm}}(B_k) + \mu_k B_k\right] + \nu \left(\sum_{k=1}^{K} B_k - B\right), \end{aligned} \tag{46}$$

Then we have the following conditions:

$$\begin{aligned} &B_k^* \geq 0,\ \sum_{k=1}^{K} B_k^* = B,\ \ \mu_k^* \geq 0,\ \mu_k^* B_k^* = 0,\ \forall k \in \mathcal{K}, \\ &\qquad\qquad \left.\frac{\partial E_k^{\text{cmm}}}{\partial B_k}\right|_{B_k^*} - \mu_k^* + \nu^* = 0, \quad \forall k \in \mathcal{K} \end{aligned} \tag{47}$$

The feasible requirement gives $B_k^* > 0, \forall k \in \mathcal{K}$, so that $\mu_k^* = 0$. Therefore, $\left.\frac{\partial E_k^{\text{cmm}}}{\partial B_k}\right|_{B_k^*} = -\nu^*,\ \forall k \in \mathcal{K}$.

Appendix E
Proof of Lemma 4

To begin with, we introduce several basic properties: 1) B_k is a decreasing convex function of ν with the form in Lemma 2; 2) ν_k is a decreasing and strictly convex function of B_k with the form in (15); 3) $\nu_k(B_k) > 0$ for all k and all value of $B_k > 0$. The proofs are straightforward and omitted for brevity.

Assume the point $\nu^{(i-1)} > \nu^*$, it follows from property 1) that $B_k(\nu^{(i-1)}) < B_k^*$, $\forall k \in \mathcal{K}$. Denote $B_k^{(i)} \triangleq B_k(\nu^{(i-1)})$, and it holds that $\sum_{k=1}^K B_k^{(i)} < \sum_{k=1}^K B_k^* = B$. Therefore, $\text{sgn}\big(\sum_{k=1}^K B_k^{(i)} - B\big) = -1$ and $\widetilde{B}_k^{(i)} = \frac{B}{\sum_{k=1}^K B_k^{(i)}} B_k^{(i)} > B_k^{(i)}$. By property 2), we know that $\nu_k(\widetilde{B}_k^{(i)}) < \nu_k(B_k^{(i)}) = \nu^{(i-1)}$, $\forall k \in \mathcal{K}$. Then, we have $\max_k\{\nu_k(\widetilde{B}_k^{(i)})\} < \nu^{(i-1)}$.

Next, we prove by contradiction that $\max_k\{\nu_k(B_k^{(i)})\} > \nu^*$ as follows. First, we assume that $\max_k\{\nu_k(\widetilde{B}_k^{(i)})\} < \nu^*$ and one can have $\nu_k(\widetilde{B}_k^{(i)}) < \nu^*,\ \forall k \in \mathcal{K}$. Accordingly, we have $\widetilde{B}_k^{(i)} > B_k^*$, $\forall k \in \mathcal{K}$, which implies that $\sum_{k=1}^K \widetilde{B}_k^{(i)} > \sum_{k=1}^K B_k^* = B$. However, it is invalid as $\sum_{k=1}^K \widetilde{B}_k^{(i)} = \sum_{k=1}^K \frac{B}{\sum_{k=1}^K B_k^{(i)}} B_k^{(i)} = \frac{B}{\sum_{k=1}^K B_k^{(i)}} \sum_{k=1}^K B_k^{(i)} = B$. Thus the earlier assumption is false. Thereby, we have $\max_k\{\nu_k(\widetilde{B}_k^{(i)})\} > \nu^*$.

Following the same procedure, we can derive for the case of $\nu^{(i-1)} < \nu^*$, that $y = \text{sgn}\left(\sum_{k=1}^K B_k^{(i)} - B\right) = 1$ and $\nu^{(i-1)} < \min_k\{\nu_k(\widetilde{B}_k^{(i)})\} < \nu^*$.

Acknowledgment

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence, or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: Wireless communication meets machine learning," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 19–25, Jan. 2020.

[2] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 491–506, Jan. 2020.

[3] M. M. Amiri and D. Gunduz, "Machine learning at the wireless edge: Distributed stochastic gradient descent Over-the-Air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, 2020.

[4] S. Wang *et al.*, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. INFOCOM-IEEE Conf. Comput. Commun.*, Honolulu, HI, USA, Apr. 2018, pp. 63–71.

[5] S. Mittal and J. S. Vetter, "A survey of CPU-GPU heterogeneous computing techniques," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 1–35, Jul. 2015.

[6] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019, pp. 1–7.

[7] K. Yang, T. Jiang, Y. Shi, and Z. Ding, “Federated learning via over-the-air computation,” *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Mar. 2020.

[8] C. Xiong, G. Y. Li, S. Zhang, Y. Chen, and S. Xu, “Energy-efficient resource allocation in OFDMA networks,” *IEEE Trans. Commun.*, vol. 60, no. 12, pp. 3767–3778, Dec. 2012.

[9] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, “Energy-efficient radio resource allocation for federated edge learning,” in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Dublin, Ireland, Jun. 2020, pp. 1–6.

[10] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, “A joint learning and communications framework for federated learning over wireless networks,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.

[11] Y. Sun, S. Zhou, and D. Gunduz, “Energy-aware analog aggregation for federated learning with redundant data,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–7.

[12] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, “Energy efficient federated learning over wireless communication networks,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, Mar. 2021.

[13] X. Mo and J. Xu, “Energy-efficient federated edge learning with joint communication and computation design,” 2020, *arXiv:2003.00199*. [Online]. Available: http://arxiv.org/abs/2003.00199

[14] A. K. Singh, K. R. Basireddy, A. Prakash, G. V. Merrett, and B. M. Al-Hashimi, “Collaborative adaptation for energy-efficient heterogeneous mobile SoCs,” *IEEE Trans. Comput.*, vol. 69, no. 2, pp. 185–197, Feb. 2020.

[15] L. W. Chang, J. G. Luna, I. E. Hajj, S. Huang, D. Chen, and W. Hwu, “Collaborative computing for heterogeneous integrated systems,” in *Proc. 8th ACM/SPEC Int. Conf. Perform. Eng.*, L’Aquila, Italy, Apr. 2017, pp. 385–388.

[16] A. Ignatov *et al.*, “AI Benchmark: Running deep neural networks on Android smartphones,” in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, Munich, Germany, Sep. 2018, pp. 288–314.

[17] Y. Kim, J. Kim, D. Chae, D. Kim, and J. Kim, “μLayer: Low latency on-device inference using cooperative single-layer acceleration and processor-friendly quantization,” in *Proc. 14th EuroSys Conf.*, Dresden, Germany, Mar. 2019, pp. 1–15.

[18] C. Chen, K. Li, A. Ouyang, and K. Li, “FlinkCL: An OpenCL-based in-memory computing architecture on heterogeneous CPU-GPU clusters for big data,” *IEEE Trans. Comput.*, vol. 67, no. 12, pp. 1765–1779, Dec. 2018.

[19] Y. Lee, K. G. Shin, and H. S. Chwa, “Thermal-aware scheduling for integrated CPUs-GPU platforms,” *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 5s, pp. 1–25, Oct. 2019.

[20] S. Naffziger. *AMD’s Commitment to Accelerating Energy Efficiency*. Accessed: Jun. 25, 2020. [Online]. Available: https://www.amd.com/system/files/documents/energy-efficiency-whitepaper.pdf

[21] J.-G. Park, C.-Y. Hsieh, N. Dutt, and S.-S. Lim, “Synergistic CPU-GPU frequency capping for energy-efficient mobile games,” *ACM Trans. Embedded Comput. Syst.*, vol. 17, no. 2, pp. 1–24, Apr. 2018.

[22] S. Rai and M. Chaudhuri, “Using criticality of GPU accesses in memory management for CPU-GPU heterogeneous multi-core processors,” *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, pp. 1–23, 2017.

[23] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. Int. Conf. Artif. Intell. Statist.*, Fort Lauderdale, FL, USA, Apr. 2017, pp. 1273–1282.

[24] X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An extremely efficient convolutional neural network for mobile devices,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, Jun. 2018, pp. 6848–6856.

[25] Z. Cui, Y. Liang, K. Rupnow, and D. Chen, “An accurate GPU performance model for effective control flow divergence optimization,” in *Proc. IEEE 26th Int. Parallel Distrib. Process. Symp.*, Shanghai, China, May 2012, pp. 83–94.

[26] J. Lee, V. Sathisha, M. Schulte, K. Compton, and N. S. Kim, “Improving throughput of power-constrained gpus using dynamic voltage/frequency and core scaling,” in *Proc. Int. Conf. Parallel Archit. Compilation Techn.*, Galveston, TX, USA, Oct. 2011, pp. 111–120.

[27] M. Liu, X. Xu, and D. Li, “Learning convolution neural networks on heterogeneous CPU-GPU platform,” U.S. Patent 010 002 402 B2, Jun. 19, 2018.

[28] A. Vega, A. Buyuktosunoglu, and P. Bose, “Transparent CPU-GPU collaboration for data-parallel kernels on heterogeneous systems,” in *Proc. 22nd Int. Conf. Parallel Archit. Compilation Techn. (PACT)*, Oct. 2013, pp. 245–255.

[29] O. Valery, P. Liu, and J. Wu, “A collaborative CPU-GPU approach for deep learning on mobile devices,” *Concurr. Comput. Pract. Exper.*, vol. 31, no. 17, pp. 1–21, 2019.

[30] C. Liu, J. Li, W. Huang, J. Rubio, E. Speight, and X. Lin, “Power-efficient time-sensitive mapping in heterogeneous systems,” in *Proc. 21st Int. Conf. Parallel Archit. Compilation Techn. (PACT)*, Minneapolis, MN, USA, 2012, pp. 23–32.

[31] Z. Wang, L. Cheng, W. Zhao, and N. Xiong, “An architecture-level graphics processing unit energy model,” *Concurrency Comput. Pract. Exper.*, vol. 28, no. 10, pp. 2795–2810, Jul. 2016.

[32] V. Kumar, S. Shekhar, and M. B. Amin, “A scalable parallel formulation of the backpropagation algorithm for hypercubes and related architectures,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 10, pp. 1073–1090, Oct. 1994.

[33] R. Rojas, *Neural Networks: A Systematic Introduction*. Berlin, Germany: Springer-Verlag, 1996.

[34] Y. Jiao, H. Lin, P. Balaji, and W. Feng, “Power and performance characterization of computational kernels on the GPU,” in *Proc. IEEE/ACM Int. Conf. Green Comput. Commun. Int. Conf. Cyber, Phys. Social Comput.*, Hangzhou, China, Dec. 2010, pp. 221–228.

[35] J. Bernstein, Y. X. Wang, K. Azizzadenesheli, and A. Anandkumar, “signSGD: Compressed optimisation for non-convex problems,” in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, Jul. 2018, pp. 560–569.

[36] D. Wen, M. Bennis, and K. Huang, “Joint parameter-and-bandwidth allocation for improving the efficiency of partitioned edge learning,” *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 8272–8286, Sep. 2020.

[37] C. You, K. Huang, H. Chae, and B.-H. Kim, “Energy-efficient resource allocation for mobile-edge computation offloading,” *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[38] S. Wang *et al.*, “Adaptive federated learning in resource constrained edge computing systems,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[39] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, “Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing,” 2020, *arXiv:2007.07122*. [Online]. Available: http://arxiv.org/abs/2007.07122

[40] A. Khaled, K. Mishchenko, and P. Richtárik, “First analysis of local GD on heterogeneous data,” 2019, *arXiv:1909.04715*. [Online]. Available: http://arxiv.org/abs/1909.04715

[41] J. Huang, V. Subramanian, R. Agrawal, and R. Berry, “Joint scheduling and resource allocation in uplink OFDM systems for broadband wireless access networks,” *IEEE J. Sel. Areas Commun.*, vol. 27, no. 2, pp. 226–234, Feb. 2009.

[42] D. Kivanc, G. Li, and H. Liu, “Computationally efficient bandwidth allocation and power control for ofdma,” *IEEE Trans. Wireless Commun.*, vol. 2, no. 6, pp. 1150–1158, Nov. 2003.

Qunsong Zeng (Graduate Student Member, IEEE) received the B.S. degree in physics from the University of Science and Technology of China (USTC). He is currently pursuing the Ph.D. degree with the Department of Electrical and Electronic Engineering, The University of Hong Kong (HKU). His research interests include edge intelligence, distributed machine learning, MIMO-OFDM baseband processing, and wireless power transfer.

Yuqing Du (Student Member, IEEE) received the B.Eng. degree from Harbin Engineering University (HEU) in 2016 and the Ph.D. degree from The University of Hong Kong (HKU) in 2020, all in electrical engineering. His research interests include edge learning and distributed machine learning.

Kaibin Huang (Fellow, IEEE) received the B.Eng. and M.Eng. degrees from the National University of Singapore and the Ph.D. degree from The University of Texas at Austin, all in electrical engineering. He is currently an Associate Professor with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong. He received the IEEE Communication Society's 2021 Best Survey Paper, the 2019 Best Tutorial Paper, the 2019 Asia–Pacific Outstanding Paper, the 2015 Asia–Pacific Best Paper Awards as well as Best Paper Awards at IEEE GLOBECOM 2006 and IEEE/CIC ICCC 2018. He also received the Outstanding Teaching Award from Yonsei University, South Korea, in 2011. He has served as the Lead Chair for the Wireless Communications Symposium of IEEE Globecom 2017 and the Communication Theory Symposium of IEEE GLOBECOM 2014 and the TPC Co-Chair for IEEE PIMRC 2017 and IEEE CTW 2013. He is an Executive Editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, an Associate Editor of IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (IEEE JSAC), and an Area Editor of IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING. He has also served on the Editorial Board for IEEE WIRELESS COMMUNICATIONS LETTERS. He has guest edited Special Issues of IEEE JSAC, IEEE JOURNAL ON SELECTED TOPICS IN SIGNAL PROCESSING, and *IEEE Communications Magazine*. He is an IEEE Distinguished Lecturer of IEEE Communications and Vehicular Technology Societies. He has been named as a Highly Cited Researcher by Clarivate Analytics in 2019 and 2020.

Kin K. Leung (Fellow, IEEE) received the B.S. degree from the Chinese University of Hong Kong in 1980 and the M.S. and Ph.D. degrees from the University of California at Los Angeles, Los Angeles, in 1982 and 1985, respectively.

He has worked with AT&T Bell Labs and its successor companies in NJ, from 1986 to 2004. Since then, he has been the Tanaka Chair Professor of the Electrical and Electronic Engineering (EEE), and Computing Departments, Imperial College London. He is currently the Head of the Communications and Signal Processing Group, EEE Department. His current research interests include optimization and machine learning for design and control of large-scale communication, computer, and sensor infrastructures. He also works on multi-antenna and cross-layer designs for wireless networks.

Dr. Leung received the Distinguished Member of Technical Staff Award from AT&T Bell Labs in 1994 and was a co-recipient of the Lanchester Prize Honorable Mention Award in 1997. He received the Royal Society Wolfson Research Merits Award from 2004 to 2009, and became a member of Academia Europaea in 2012 and an IET Fellow in 2021. Jointly with his co-authors, he received the IEEE ComSoc Leonard G. Abraham Prize in 2021 and Best Paper Awards at the IEEE PIMRC 2012, ICDCS 2013, and ICC 2019. He has served as a member (2009–2011) and the Chairman (2012–2015) of the IEEE Fellow Evaluation Committee for ComSoc. He has served as an Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC): Wireless Series, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and IEEE TRANSACTIONS ON COMMUNICATIONS. He is the Chair of the Steering Committee of the IEEE TRANSACTIONS ON MOBILE COMPUTING. He is an Editor of the *ACM Computing Surveys* and *International Journal on Sensor Networks*.